A Practical Performance Comparison of Parallel Matrix Multiplication Algorithms on Network of Workstations

Kalim Qureshi* Non-member
Haroon Rashid** Non-member

In this paper we practically compared the performance of three blocked based parallel multiplication algorithms with simple fixed size runtime task scheduling strategy on homogeneous cluster of workstations. Parallel Virtual Machine (PVM) was used for this study.

Keywords : Parallel algorithms, Load balancing, matrix multiplication, performance analysis, cluster parallel computing

1. Introduction

Matrix multiplication is a problem which has a wide application in many fields of science. But the problem costs too much computational complexity if it is solved using the conventional serial algorithm. There are some serial algorithms like⁽¹⁾ which may improve the performance a bit. To gain much more performance, the parallel algorithm implemented on distributed system is the best solution. As the matrix multiplication is embarrassingly parallel, so this application is relatively easy to implement. But it is still an issue of great interest to develop a matrix multiplication algorithm computationally optimal. In this paper, the four existing parallel matrix multiplication algorithms will be implemented and their performance from many aspects will be measured. The algorithms are RTS (Run Time Strategy)⁽²⁾, simple block based algorithm⁽³⁾, Cannon's Algorithm and Fox's Algorithm⁽³⁾. RTS has been chosen as it is considered one of the best non block based simple parallel matrix multiplication algorithms.

2. Investigated Parallel Algorithms on Network of Workstation (NOW)

2.1 Runtime Task Scheduling (RTS) Strategy In RTS strategy, a unit of task (one row and one column) is distributed at runtime. As the node completes the previous assigned sub-task, new task is assigned for processing by the master. The number of task processed by the nodes is highly depends on the node's performance. It cope the heterogeneity and load imbalance of nodes. The size of the task plays a great role of performance under this strategy.

2.2 Block based Matrix Multiplication The concept of block matrix multiplication will be used for all the rest three algorithms. For example, one $n \times n$ matrix A can be regarded as $q \times q$ array of blocks such that each block is an $(n\sqrt{q}) \times (n\sqrt{q})$ sub-matrix. We can use p processors to implement the block version of matrix multiplication algorithm in parallel, where $q=\sqrt{p}$. This algorithm can be easily parallelized if an all-to-all broadcast of matrix A 's blocks is performed in each row of processors and an

all-to-all broadcast of matrix B 's blocks is performed in each column.

The serial block based matrix multiplication algorithm is below:

Procedure BLOCK_MAT_MUL (A,B,C)

Begin

For i=0 to q-1

For j= 0 to q-1 do

Begin

Initialize all elements of $C_{i,j}$ to zero

For k=0 to q-1 do

$C_{i,j} = C_{i,j} + A_{i,k} \times B_{k,j}$;

Endfor

End BLOCK_MAT_MUL

2.3 Cannon's Algorithm The matrices A and B are partitioned into p square blocks. We label the processors from $P(0,0)$ to $P(\sqrt{p}-1, \sqrt{p}-1)$. Initially $A(i, j)$ and $B(i, j)$ are assigned to $P(i, j)$. The blocks are systematically rotated among the processors after every sub-matrix multiplication so that every processor gets a fresh $A(i,k)$ after each rotation Cannon's algorithm reorders the summation in the inner loop of block matrix multiplication as follows:

$$\begin{aligned} C(i,j) &= C(i,j) + \sum_{k=0}^{s-1} A(i,k) * B(k,j) \\ &= C(i,j) + \sum_{k=0}^{s-1} A(i, (i+j+k) \bmod s) * \\ &\quad B((i+j+k) \bmod s, j) \end{aligned}$$

Cannon's matrix multiplication algorithm

for all (i=0 to s-1) ... "skew" A

Left-circular-shift row i of A by i,

so that $A(i,j)$ is overwritten by $A(i, (j+i) \bmod s)$

end for

for all (i=0 to s-1) ... "skew" B

Up-circular-shift column i of B by i,

so that $B(i,j)$ is overwritten by $B((i+j) \bmod s, j)$

end for

for k=0 to s-1

for all (i=0 to s-1, j=0 to s-1)

$C(i,j) = C(i,j) + A(i,j) * B(i,j)$

Left-circular-shift each row of A by 1,

so that $A(i,j)$ is overwritten by $A(i, (j+1) \bmod s)$

Up-circular-shift each column of B by 1,

so that $B(i,j)$ is overwritten by $B((i+1) \bmod s, j)$

end for

* Math. And Computer Science Department, Kuwait University, Kuwait.

** COMSATS Institute of Information Technology, Abbottabad, Pakistan.

end for

2.4 Fox's Algorithm This is another memory efficient parallel algorithm for multiplying dense matrices. Again, both $n \times n$ matrices A and B are partitioned among p processors so that each processor initially stores $(n/p) \times (n/p)$ blocks of each matrix. The algorithm uses one to all broadcasts of the block blocks of matrix A in processor rows, and single step circular upward shifts of the blocks of matrix B along processor columns. Initially each diagonal block $A(i, j)$ is selected for broadcast.

The pseudo-code of Fox's algorithm is given below:

```
// Partition the matrices A and B into sub-matrices and send to the
processors
node = 0
FOR i = 1, m/sqrt(N)
  FOR j = 1, m/sqrt(N)
    ii = 0
    FOR k = [(i - 1)m/sqrt(N)] + 1, im/sqrt(N)
      ii = ii + 1
      jj = 0
      FOR l = [(j - 1)m/sqrt(N)] + 1,
        jm/sqrt(N)
        jj = jj + 1
        Aii,jj = akl // akl represents the
        element in the kth
        // row and lth column of
        A, whereas
        // Aii,jj represents a
        sub-matrix of A
        Bii,jj = bkl // as above but for B
      ENDFOR
    ENDFOR
    SEND m, N, i and j to current node
    SEND Aij to current node
    SEND Bij to current node

    node = node + 1
  ENDFOR
ENDFOR
// Wait for the node to compute the sub-matrix products
WAIT for nodes to finish
// Read sub-matrices Cij from the nodes
FOR node = 0, N - 1
  READ Cij from current node
ENDFOR
// Assemble C from all the Cij's
OUTPUT C
END PROGRAM
```

3. Implementation and Measurement

These entire four algorithms are implemented in UNIX environment on SUN homogenous cluster of machines. Also, workstations are interconnected by an Ethernet 10 Mb/s LAN and PVM library has been used to create process and distribute the tasks.

The algorithms were tested for 16×16 and 64×64 size of matrices. All these results are summarized in Figure 1.

From Figure 1 it is evident that any block based matrix multiplication algorithm is far better than simple matrix multiplication algorithms. The Cannon's and Fox's algorithms' performances are almost same. But they both are slightly better

Fig. 1. The measure processing time of four algorithms using matrix of 64×64 and 16×16

Fig. 2. Memory used by all four algorithms

than the simple block based algorithm.

The main facility of Cannon's and Fox's algorithm is the memory usage per process is very less than the other existing algorithms. This is very prominent from Figure 2.

(Manuscript received April 1, 2004)

References

- (1) D. H. Bailey, K. Lee, and H. D. Simon: "Using Strassen's Algorithm to Accelerate the Solution of Linear Systems", Journal of Supercomputing, Vol.4, No.4, p.357-371 (Jan. 1991)
- (2) K. Qureshi and M. Hatanaka: "An empirical study of task scheduling strategies for image processing application on heterogeneous distributed computing system", special issue of the Parallel Distributed Computing Practices Journal, Vol.3, No.3, pp.297-306 (Spt. 2000)
- (3) A. Grama, A. Gupta, G. Karypis, and V. Kumar: Introduction to Parallel Computing, Second, Addison Wesley, ISBN: 0-201-64865-2, pp.345-349 (2002)

Kalim Qureshi (Non-member) He is an Assistant Professor in Department of Math and Computer Science at Kuwait University, Kuwait. He is did his MS and Ph.D from Dept. of Computer Science and Systems Engg., Muroran Institute of Technology, Hokkaido, Japan in 1997 and 2000 respectively. He is a member of IEEE Computer Society. He can be contact by E-mail: qureshi@sci.kuniv.edu.kw

Haroon Rashid (Non-member) is an Associate Professor in Department of Computer Science at COMSATS Institute of Information Technology, Abbottabad Campus, Pakistan. His research interests include parallel computing, distributed systems, high speed networks, multimedia, and network performance optimization. His E-mail: haroon@ciit.net.pk